(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 September 2017 (21.09.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/160309 A1

(51) International Patent Classification:
G06F 9/54 (2006.01) *G06F 17/30* (2006.01)
G06F 21/50 (2013.01)

(21) International Application Number:
PCT/US2016/023120

(22) International Filing Date:
18 March 2016 (18.03.2016)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **ENTIT SOFTWARE LLC** [US/US]; 1140 Enterprise Way, Building G, Sunnyvale, CA 94089 (US).

(72) Inventors: **NG, Ming Sum Sam**; 1111 King's Road, Taikoo Shing, 19/Fl., Cityplaza One, Hong Kong (CN). **MUTHURAJAN, Sasi Siddharth**; 150 Cambridge Park Drive, Cambridge, Massachusetts 02140 (US). **RAZ, Barak**; 1160 Enterprise Way, Sunnyvale, California 94089 (US).

(74) Agents: **PINKSTON, Caroline** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop #79, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

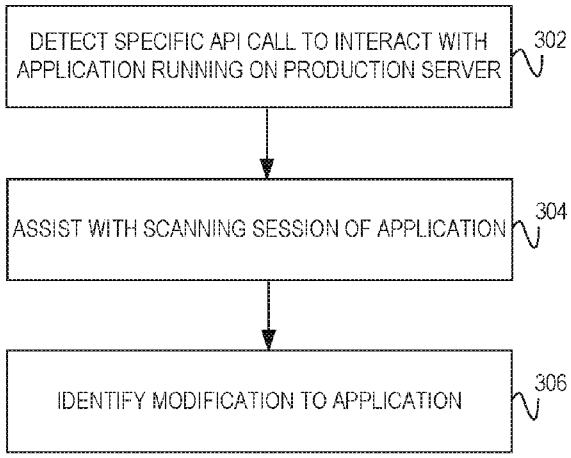
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

(54) Title: ASSISTING A SCANNING SESSION

FIG. 3



(57) Abstract: Examples herein disclose via use of a physical processor, detecting a specific application programming interface (API) call to interact with an application running on a production server. Based on the detection of the specific API call, the examples assist, using the physical processor, a scanning session based on the specific API call. Using the physical processor, the examples identify a modification to the application based on the scanning session.

WO 2017/160309 A1

ASSISTING A SCANNING SESSION

BACKGROUND

[0001] A web application is a software application which is hosted and executed by one or more production servers to run in a web browser. Devices communication with the web application using application programming interfaces (APIs). An API comprises a set of API calls to interact with the web application.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] In the accompanying drawings, like numerals refer to like components or blocks. The following detailed description references the drawings, wherein:

[0003] FIG. 1 is a block diagram of an example system including a scanner and production server to interact with an application for detection of a specific API call in accordance with the present disclosure;

[0004] FIG. 2 is an illustration of example system including a scanner and a production server to interact with an application in accordance with the present disclosure;

[0005] FIG. 3 is a flowchart of an example method executable by a computing device to assist with a scanning session for identification of a modification to an application in accordance with the present disclosure;

[0006] FIG. 4 is a flowchart of an example method executable by a computing device to assist with a scanning session of an application based on a detected API call in accordance with the present disclosure;

[0007] FIG. 5 is a block diagram of an example computing device with a processing resource to execute instructions in a machine-readable storage medium for detecting a modification to an application in accordance with the present disclosure; and

[0008] FIG. 6 is a block diagram of an example computing device with a processing resource to execute instructions in a machine-readable storage medium for assisting with a scanning session of an application for detection of a modification to the application in accordance with the present disclosure.

DETAILED DESCRIPTION

[0009] A web application, also referred to as the application, may be susceptible to security attacks which may lead to a compromised application. Web application scanning is a process to identify potential security vulnerabilities in the application. The potential security vulnerabilities, as discussed herein, may include an issue related to the application which can cause security implications. Examples of potential security vulnerabilities may include structured query language (SQL) injection, cross-site scripting, modification of source code, modification of data, etc.

[0010] There are several side effects to the scanning process including unwanted modifying of the production environment and triggering unnecessary tasks. The first side effect, modification of the production environment, may include leaving garbage data in a database, modifying a data file during the scanning process, affecting data seen by a user of the application, etc. For example, scanning a live blogging web application which may leave thousands of spam like comments viewable by users of the blog. The second side effect, triggering unnecessary tasks, may include, for example, generating a larger number of email notifications and posting unnecessary data to a backend of a server, among others. These side effects may slow down the processing of the application and adversely affect the usability of the application.

[0011] To address these issues, examples disclosed herein can detect a potential security vulnerability during a scanning session without affecting the processing and usability of the application. As such, the examples detect a particular API call to interact with the application running on a production server. Based on the detection of the particular API call, the examples assist with a scanning session of the application. By assisting with the scanning session of the application, the examples may identify a modification to the application which may represent a potential security vulnerability. By monitoring and detecting the particular API calls, the impact incurred on the production server may be minimized without adversely affecting functionality of the scanning session.

[0012] Additionally, the examples allow a dynamic scanning session to detect the potential security vulnerabilities in the production environment while minimizing disruption to the data of the application.

[0013] The following detailed description refers to the accompanied figures. Wherever possible, the same reference numbers are used in the figures and the following description to refer to the same or similar parts. It is to be expressly understood, however, that the figures

are for the purpose of illustration and description only. While several examples are described throughout, modification, adaptions, and other implementations are possible. Accordingly, the following detailed description is not meant to limit the disclosed examples, rather it is meant to provide proper scope of the disclosed examples and may be defined by the appended claims.

[0014] FIG. 1 illustrates an example system including a scanner 102 and a production server 104. The production server 104 hosts and runs application 106. The system in FIG. 1 illustrates a production environment that could include a network of multiple geographically distributed machines in data centers, virtual machines, other devices that facilitate the functionality described herein, or any combination thereof. Although FIG. 1 illustrates components 102, 106, and 108, implementations should not be limited as the system may also include a database, redundant resources, and/or other resources.

[0015] The scanner 102 transmits a signal to the production server 104 for commencement of a scanning session. The signal may be in the form of a data packet, may be an electronic signal, and/or other type of signal. For example, a signal in the form of a data packet may include information identifying the scanner 102 as authorized to perform the scanning session to the security agent 108. The data packet may include information such as an internet protocol (IP) address which identifies the scanner 102. In this regard, the scanner 102 provides the authorization to the security agent 108 for performing the scanning session on the application 106.

[0016] The scanner 102 may implement a scanning session by performing a dynamic analysis on the application 106 by emulating an attacker. For example, the scanner 102 may emulate an attacker by sending attacks over a network connection to application 106. The security agent 108, in turn, may monitor the behavior of the application 106 to identify any modifications which may indicate a vulnerability to the scanner 102. In particular, during the the scanning session, the security agent 108 may track various API calls which interact with application 106 to determine if any modifications have occurred. A modification may represent an unexpected behavior of the application 106 which may signal to the scanner 102 a potential vulnerability. The security agent 108 may communicate information about the identified modifications to the scanner 102. The scanner 102 may use this information to identify the potential vulnerability and improve the quality and performance of the scanning session.

[0017] The security agent 108 may track various API calls by monitoring API calls executed by a runtime agent for the application. The security agent 108 may determine whether an individual API call warrants assisted scanning. Based on detection of the specific API call, the

security agent 108 proceeds to assist with the scanning session for identification of a modification.

[0018] In some examples, the security agent 108 may determine whether an individual API call warrants assisted scanning by determining whether a particular API call is associated with assisting scanning. For example, a non-transitory storage medium communicably coupled to the security agent 108 and/or the production server 102 may comprise a list of APIs that warrant assisting scanning. In some examples, the list may be pre-defined, may be obtained from an administrator of the application and/or the production server 102, may be customized for the application, may be machine learned based on behaviors of the application, may be determined based on any combination thereof, and/or may otherwise be determined. In some examples, the list of APIs may comprise APIs related to file modification, database connection, related to email, etc.

[0019] The scanner 102 may comprise a combination of hardware and programming that may be running on a different server from the production server 104 and as such may include instructions (e.g., stored on a machine-readable medium) that, when executed (e.g., by the different server), implement the functionality of the scanner 102. Alternatively, or in addition, scanner 102 may include electronic circuitry (i.e., hardware) that implements the functionality of scanner 102. Although illustrated as a separate component to production server 104, this was done for purposes of illustration as the scanner 102 may be considered part of production server 104.

[0020] The production server 104, as part of the production environment, hosts, executes, and runs application 104. Implementations of production server 104 include, by way of example, a Local Area Network (LAN) server, web server, application server, cloud server, network server, file server, data server, or other type of computing device capable hosting and running the application 106.

[0021] The application 106 is a program or group of programs designed for user(s) interaction. For example, application 106 may comprise a web-based application to refer to the program or group of programs being accessed over a network connection, such as hypertext text transfer protocol (HTTP). Implementations of the application 106 include, by way of example, content access software, media development software, educational software, information worker software, business software, enterprise infrastructure software, enterprise software, application suite, among others.

[0022] The security agent 108 is an engine residing at the production server 104. The security agent 108, based on the performance of the scanning session by the scanner 102, monitors various API calls. The security agent 108 may include, by way of example, instructions (e.g., stored on a machine-readable medium) that, when executed (e.g., by a processor and/or controller), implement the functionality of security agent 108. Alternatively, or in addition, the security agent 108 may include electronic circuitry (i.e., hardware) that implements the functionality of security agent 108 to detect the specific API call and assist with the scanning session for identification of the modification to application 106.

[0023] The security agent 108 may perform functionality set forth in engines 110-114. At engines 110-114, the security agent 108 detects the specific API call, the detection of the specific API call signals for the security agent 108 to assist with the scanning session for identification of the modification within the application 106. Specifically, at engine 110 the security agent 108 tracks various API calls which interact with the application 106 for detection of the specific API call.

[0024] The process of assisting with the scanning session may be dependent on the specific API call detected at engine 110. Alternatively, the assistance of the scanning session may be related to the functionality of the detected API call. In implementations, the detected API call may be related to a file modification, a database connection, and/or email. The process for assisting with the scanning session may depend on one of the specific listed API calls. For example, the specific API call may include the file modification API. In this example, the assistance of the scanning session may include skipping the execution of the file modification API call and reporting the skipped execution file modification API call to the scanner 102. In an alternate example, the assistance of the scanning session may include backing up a file affected by the file modification API call, executing the file modification API call, and reverting the modified file to a state prior to execution of the API call. In another example, the specific API may be related to the database connection. In this example, the database connection API call may be redirected from an originally intended database to a testing database for execution. In an alternative example, the database connection API call is executed which modifies the intended database and the modified database is reverted to a state prior to execution of the API call. In a further example, the specific API call may be related to email. In this example, the email API call may be intercepted and redirected from the originally intended email address to a different email address.

[0025] At engine 114, during the scanning session performed by the scanner 102, the security agent 108 identifies the modification to application 106. Responsive to this identification, the security agent 108 can alert the scanner 102 to the modification. Implementations of engines 110-114 may include, by way of example, instructions (e.g., stored on a machine-readable medium) that, when executed (e.g., by the security agent 108), implement the functionality of engines 110-114. Alternatively, or in addition, the engines 110-114 may include electronic circuitry (i.e., hardware) that implements the functionality of engines 110-114.

[0026] FIG. 2 illustrates an example system including a scanner 102 to communicate with security agent 108 to perform a scanning session on application 106 running on production server 104. Based on the scanner 102 performing the scanning session of the application 106, the security agent 108 detects a specific API call to interact with the application 106. Based on the detection of this specific API call, the security agent 108 proceeds to assist with the scanning session. In one implementation, the assistance of the scanning session by the security agent 108 includes intercepting the specific API call and transmitting the API call to a redundant resource, such as a test database 220 for execution of the specific API call. In this implementation, the security agent 108 redirects the specific API from the originally intended resource, such as database 218 that would originally execute the specific API call, to the redundant resource, such as test database 220.

[0027] Test database 220 may comprise a redundant resource for execution of the detected specific API. Implementations of the redundant resource include a virtual component, physical component, or combination thereof for execution of the detected specific API. In one implementation, the security agent 108 copies at least a portion of database 218 upon the detection of the specific API call. In this implementation, the security agent 108 maintains a mapping list of addresses to the original intended resource and the redundant resource so the specific API call may be routed, accordingly. Thus upon the detection of the specific API call, the security agent 108 references the mapping list to identify the address of the redundant resource for execution of the specific API call.

[0028] A runtime agent 216 executes application 106, outside of the scanning session performed by scanner 102. The runtime agent 216 executes source code of the application 106 to provide the functionality of the application 106 to user(s). In that capacity, the runtime agent 216 may execute API calls to execute functionality of the application 106. The security agent 108 may intercept API calls executed by the runtime agent 216 to determine whether to assist

with scanning. Implementations of the runtime agent 216 may include, by way of example, instructions (e.g., stored on a machine-readable medium) that, when executed (e.g., by a processor and/or controller), implement the functionality of runtime agent 216. Alternatively, or in addition, the runtime agent 216 may include electronic circuitry (i.e., hardware) that implements the functionality of 216.

[0029] Referring now to FIGS. 3 and 4, example flowcharts are illustrated in accordance with various examples of the present disclosure. The flowcharts represent processes that may be utilized in conjunction with various systems and devices as discussed with reference to the preceding figures. While illustrated in a particular order, the flowcharts are not intended to be so limited. Rather, it is expressly contemplated that various processes may occur in different orders and/or simultaneously with other processes than those illustrated.

[0030] FIG. 3 is a flowchart of an example method to identify a modification to an application running on a production server. The method is executable by a computing device for identification of the modification. The computing device detects a specific API call among various API calls which interact with the application running on the production server. Upon the detection of the specific API call, the computing device assists with a scanning session of the application. Based on the scanning session, the computing device proceeds to identify the modification to the application. In discussing FIG. 3, references may be made to the components in FIGS. 1-2 to provide contextual examples. In one implementation, the security agent 108 executes operations 302-306 to detect the specific API call. Although FIG. 3 is described as implemented by the security agent 108, it may be executable on other suitable hardware components. For example, FIG. 3 may be implemented in the form of executable instructions on a machine-readable storage medium 504 and 604 as in FIGS. 5-6.

[0031] At operation 302, the computing device monitors various API calls interacting with the application running on the production server. By monitoring the various API calls, the computing device detects the specific API call among the various API calls. The specific API call is a predetermined API call which signals to the computing device to assist with the scanning session. The computing device monitors the various API calls which are used to build and execute the application. In this manner, the computing device tracks each of the various API calls prior to execution. This allows the computing device to determine which API call is slated for execution. Responsive to determining the API call slated for execution, the computing device detects the specific API call which alerts the computing device to perform assistance with the scanning session. The API calls may include a set of routines, protocols,

tools, and/or resources related to the execution and establishment of the application. As such, each of the API calls may be related to different functionality of the application. In implementations, the detected specific API call may be related to a file modification, database connection, email, etc. Each of these detected specific API calls represent a different type of API call. Depending on the detected specific API call, the computing device proceeds to assist the scanning session as at operation 304.

[0032] At operation 304, based on the detection of the specific API call, the computing device assists with the scanning session. The scanning session is initiated by a scanner prior to detection the specific API call. During the scanning session, various API calls may interact with the application. The computing device identifies the specific API call among the various API calls to proceed with assisting with the scanning session. Based on the assistance with the scanning session, the computing device identifies the modification to the application. The scanning session (as initiated by the scanner) may not have access to the application source code and thus detects the potential vulnerabilities by emulating an attack on the application. The course of action for performing the assisting the scanning session may be dependent on the type of the detected specific API call. In this implementation, the type of specific API call may be related to email, database connection, file modification, etc. Each of the types of specific API calls implement a different course of action. This implementation is discussed in detail in the next figure.

[0033] At operation 306, the computing device identifies the modification to the application based on the scanning session at operation 304. The modification to the application is an unexpected change in the data of the application and/or component in a production environment which hosts and runs the application. In a more specific implementation, the modification includes an unexpected behavior of the application. In this implementation, the computing device monitors and tracks the behaviors of the application. Thus by tracking the behaviors of the application, the computing device identifies an unexpected or unpredicted behavior. Identifying the unexpected or unpredicted behavior, the computing device identifies an anomaly in the application. Responsive to identification of the anomaly in the application, the computing device determines whether the modification is considered a security vulnerability for taking proactive measures against.

[0034] FIG. 4 is a flowchart of an example method to assist with a scanning session of an application based on a detected specific API call. The method is executable by a computing device to perform the assistance with the scanning session. The scanning session is

implemented by a scanner transmitting a signal authorization which indicates to the computing device, a scanning session may commence. The authorization may occur prior to the scanning session or during the scanning session. Upon receiving the signal authorization, the computing device monitors various API calls which interact with the application to detect a specific API call. Based on the detection the specific API call, the computing device proceeds to assist with the scanning session. Based on the detected API call, the computing device assists the scanner with the scanning session by performing at least one of the sets of operations 410-414, 416-422, 424-426, 428-434, and 436-438. For example, responsive to determining that the API call is related to file modification, the computing device proceeds to at least one of the sets of operations 410-414 or 416-422. Responsive to determining the API call is related to a database connection, the computing device proceeds to at least one of the sets operations 424-426 or 428-434. Responsive to determining the API call is related to an email, the computing device proceeds to the set of operations 436-438. The set of operations in which the computing device proceeds to depends on the specific API call to interact with the application and also with the functionalities the computing device may be capable of supporting. For example, responsive to the specific API call including the file modification, the computing device may have limited storage functionality. Thus, the computing device proceeds to the set of operations 410-414 may skip and report the file modification API. Although FIG. 4 illustrates API calls related to file modification, a database connection, and/or email, implementations are not meant to be limiting as the API calls may be related to a network connection among others. In discussing FIG. 4, references may be made to the components in FIGS. 1-2 to provide contextual examples. In one implementation, the security agent 108 executes a combination of operations 402-440 to perform the assistance of the scanning session. Although FIG. 4 is described as implemented by the security agent 108, it may be executable on other suitable components. For example, FIG. 4 may be implemented in the form of executable instructions on a machine-readable storage medium 504 and 604 as in FIGS. 5-6.

[0035] At operation 402, the computing device receives an authorization of the scanning session from a scanner. For example, the computing device may include the security agent 108 and scanner 102 as in FIG. 1. In one implementation, a scanner implements a scanning session and transmits a signal to the computing device of the scanning session. The computing device receives the communication from the scanner and may determine whether the scanning session is authorized based on header information and/or body information in the signal. For example, assume the signal is communicated as a data packet, then the computing device identifies

whether the scanner is authorized to implement the scanning session of the application. The authorization may be based on the internet protocol (IP) address, a uniform resource locator (URL), or a web address of a source, destination, or other type of information identifying the scanner. Using this information signals to the computing device, the scanning session can commence. The computing device receives the authorization of the commencement of the scanning session prior to detecting the API call as at operation 404.

[0036] At operation 404, based on the authorization of the commencement of the scanning session, the computing device monitors various API calls interacting with the application. Monitoring the various API calls, the computing device can detect the specific API call. The specific API call is the particular API call among the various API calls which indicates to the computing device to assist with the scanning session of the application. The specific API call may be predefined as the API call that signals to the computing device to assist with the scanning session. In this manner, the specific API call can act as a flag to alert the computing device to assist with the scanning session. In one implementation, if the computing device does not detect the specific API call, the computing device may continue monitoring the various API calls interacting with the application. In this implementation, the non-detection of the specific API call indicates to the computing device to not assist with f the scanning session and proceeds to operation 406. Responsive to the detection of the specific API call, the computing device may proceed to assist with the scanning session at operation 408. Proceeding to assist with the scanning session of the application, the computing device may proceed to one of the operations 408, 416, 428, and 436 based on the detected API call. For example, responsive to an API call related to a file modification, the computing device proceeds to operations 410 or 416. Responsive to the API call related to a database connection, the computing device proceeds to operations 424 or 428. Responsive to the API call as related to email, the computing device proceeds to operation 436. Operation 404 may be similar in functionality to operation 302 as in FIG. 3.

[0037] At operation 406, based upon the non-detection of the specific API call, the computing devices proceeds to monitor the various API calls interacting with the application. Based on the computing device continuing to monitor the various API calls, the computing device does not assist with the scanning session of the application. In this implementation, the scanning session commences, thus the computing device continues to monitor the various API calls, but the computing device may not assist with the scanning session.

[0038] At operation 408, the computing device proceeds to assist with the scanning session of the application based on the detection of the specific API call. The process of assisting the scanning session depends on the specific API call detected at operation 404 and as such the computing device proceeds to any set of operations 410-414, 416-422, 424-426, 428-434, 436-438, or any combination thereof. Operation 408 may be similar in functionality to operation 304 as in FIG. 3.

[0039] At operations 410-414, based on the specific API call as related to the file modification, the assistance to the scanning session may include skipping execution of the API call and reporting the skipped API call to the scanner. The file modification API call are those API calls which cause the application to modify a file. Implementations of the file modification API call include, by way of example, file create, file delete, file upload, file output, file write, etc. Table 1 lists a few examples of the file modification API calls in a different programming language. The parenthesis after each programming language calls the file which is to be modified. Although Table 1 illustrates Java and C# .NET, implementations should not be limited as this was done for purposes of clarifying examples of the present and disclosure and is not meant to limit the API calls to the illustrated programming languages.

	Java	.NET
File Create	java.io.File.createNewFile()	System.IO.StreamWriter()
File Delete	java.io.File.delete()	System.IO.File.Delete()
File Write	java.io.FileOutputStream() java.io.FileOutputStream.write()	System.IO.FileStream() System.IO.FileStream.Write()

TABLE 1

[0040] The computing device may maintain a list of the modified files according to the file modification API calls. Thus, the computing device skips or does not execute the file modification API calls. In this implementation, data of the file modification is reported back to the scanner. This enables the scanner to identify whether the file modification may cause a vulnerability issue.

[0041] At operations 416-422, in response to specific API call as related to the file modification, assistance of the scanning session includes backing up the file prior to modification, executing the file modification API, and reverting the file to the state prior to modification. At operations 416-422, a unique identification is assigned to each file

modification. Each unique identification may also be transmitted to the scanner. If the unique identification(s) are transmitted to the scanner, the scanner may decide when to revert the file to the state prior to modification. In an implementation the reversion of the file to the state prior to the execution of the API call is carried out when a session or timer expires. If the scanner decides to revert the file at a later time, care is taken to determine if the file has been additionally modified by another entity before reverting the file. In this implementation, the file hash or checksum is calculated prior to reversion to ensure the file has not been modified between the times.

[0042] At operations 424-426, in response to the specific API call as related to the database connection of the application, the computing device proceeds to direct the specific API call to a testing database. The computing device intercepts the specific API call which opens a database connection and redirects the database connection to the testing database. For example, assume the scanning session is in process, thus the scanner may transmit the request for the scanning session, while the computing device calls the API call to open the database connection. The computing device intercepts this API call to open the database connection and redirects the API call for execution at the testing database. In this manner, the testing database serves as a database redundant to the originally intended database for execution of the API. Redirecting the API call for execution at the testing database, the computing device may intercept the API call and modify the URL to redirect to the testing database. The URL of the testing database may either be provided by a user as a configurable parameter or the testing database may be created by copying the database from the production server upon the detection of the API call. Additionally, in this implementation, the computing device maintains a mapping list to the originally intended database URL and the testing database URL.

[0043] At operation 428-434, in response to the specific API call as related to the database connection, the computing device executes the API call and reverts the database to a state prior to execution of the API. The reversion of the database may be dependent on whether the computing device supports the scanning session commenced by the scanner. For example, the computing device may execute the database connection API which includes modifying data within the database. Upon modification, the computing device reverts back to the state of the database prior to execution of the database connection API. In another example, the computing device intercepts the database connection API call(s), and assigns a unique number to the database connection. This unique number is transmitted to the scanner. Thus, the API calls which close the database connection, such as close() are skipped. The scanner may signal to

the computing device when to revert the database back to the state prior to execution of the API call. Reverting the database to the state prior to execution of the specific API call may also be referred to as rolling back.

[0044] Alternatively, if the specific API call includes the database connection API, the specific API call is skipped. Skipping the API call means not executing the specific API call. Thus, the skipped API call is reported back to the scanner for further analysis.

[0045] At operation 436-438, if the specific API is related to email, the computing device proceeds to direct the API call to a different email address. For example, when testing a "contact" form, the computing device can change a recipient address from info@domain.com to info-scanning@domain.com. The mapping between the original email address and the redirected email address may be a configurable parameter by a user, developer, administrator, etc. Alternatively, the computing device intercepts a low level network connection and redirects the network connection to another local port. For example, when the application is attempting to connect to mail.domain.com:25 to send email, the computing device intercepts the email connection and redirects to a different port. Thus, the computing device may drop the connection upon receipt of the email. In a further implementation, if the specific API is related to email, the computing device can intercept the email related API and skip execution of the API while reporting the email related API back to the scanner.

[0046] In other API call implementations, the computing device monitors the network activities and/or command executions. Thus, the computing device may decide whether to skip or fail execution of the API call and report the non-executed API call to the scanner. For example, the computing device may be configured to fail the execution and returns an error when dealing with an out-of-bound network connection to backend.domain.com:443 is made. This disables the scanning of the corresponding page, but allows a reconfiguration such that the execution of the network connection may be made at a later time.

[0047] At operation 440, the computing device identifies the modification to the application based on the scanning session. Based on performing one of the sets of operations 410-414, 416-422, 424-426, 428-434, or 436-438 corresponding the specific API call, the computing device identifies the modification to the application. Operation 440 may be similar in functionality to operation 306 as in FIG. 3.

[0048] FIG. 5 is a block diagram of computing device 500 with a processing resource 502 to execute instructions 506-512 within a machine-readable storage medium 504. Although the computing device 500 includes processing resource 502 and machine-readable storage medium

504, it may also include other components that would be suitable to one skilled in the art. For example, the computing device 500 may include a controller, memory storage, or other suitable type of component. The computing device 500 is an electronic device with the processing resource 502 capable of executing instructions 506-512 and as such embodiments of the computing device 500 include a networking device, server, switch, router, mobile device, desktop computer, or other type of electronic device capable of executing instructions 506-512. The instructions 506-512 may be implemented as methods, functions, operations, and other processes implemented as machine-readable instructions stored on the storage medium 504, which may be non-transitory, such as hardware storage devices (e.g., random access memory (RAM), read only memory (ROM), erasable programmable ROM, electrically erasable ROM, hard drives, and flash memory).

[0049] The processing resource 502 may fetch, decode, and execute instructions 506-512 detect a specific API call and assist in a process of a scanning session. Specifically, the processing resource 502 executes instructions 506-512 to: authorize the scanning session; monitor various API calls which interact with the application for detection of the specific API call among the various API calls; assist with the scanning session of the application based on the detection of the specific API call; and detect the modification to the application.

[0050] The machine-readable storage medium 504 includes instructions 506-512 for the processing resource 502 to fetch, decode, and execute. In another embodiment, the machine-readable storage medium 504 may be an electronic, magnetic, optical, memory, storage, flash-drive, or other physical device that contains or stores executable instructions. Thus, the machine-readable storage medium 504 may include, for example, Random Access Memory (RAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), a storage drive, a memory cache, network storage, a Compact Disc Read Only Memory (CDROM) and the like. As such, the machine-readable storage medium 504 may include an application and/or firmware which can be utilized independently and/or in conjunction with the processing resource 502 to fetch, decode, and/or execute instructions of the machine-readable storage medium 504. The application and/or firmware may be stored on the machine-readable storage medium 504 and/or stored on another location of the computing device 500.

[0051] FIG. 6 is a block diagram of computing device 600 with a processing resource 602 to execute instructions 606-620 within a machine-readable storage medium 604. Specifically, the computing device 600 with the processing resource 602 is to detect a modification to an application based on assistance to a scanning session. Although the computing device 600

includes processing resource 602 and machine-readable storage medium 604, it may also include other components that would be suitable to one skilled in the art. For example, the computing device 600 may include a controller, memory storage, or other suitable type of component. The computing device 600 is an electronic device with the processing resource 602 capable of executing instructions 606-620 and as such embodiments of the computing device 600 include a networking device, server, switch, router, mobile device, desktop computer, laptop, or other type of electronic device capable of executing instructions 606-620. The instructions 606-620 may be implemented as methods, functions, operations, and other processes implemented as machine-readable instructions stored on the storage medium 604, which may be non-transitory, such as hardware storage devices (e.g., random access memory (RAM), read only memory (ROM), erasable programmable ROM, electrically erasable ROM, hard drives, and flash memory).

[0052] The processing resource 602 may fetch, decode, and execute instructions 606-620 to detect the modification to the application. Specifically, the processing resource 602 executes instructions 606-620 to: authorize performance of a scanning session; monitor various API calls for the detection of a specific API call; assist the scanning session depending on the specific API call detected, for example, the processor 602 may fetch and execute instruction 612 or instructions 614-616; direct execution of the specific API call to a redundant resource; skip execution of the specific API call and to report the skipped API call; and detect modification of the application.

[0053] The machine-readable storage medium 604 includes instructions 606-620 for the processing resource 602 to fetch, decode, and execute. In another embodiment, the machine-readable storage medium 604 may be an electronic, magnetic, optical, memory, storage, flash-drive, or other physical device that contains or stores executable instructions. Thus, the machine-readable storage medium 604 may include, for example, Random Access Memory (RAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), a storage drive, a memory cache, network storage, a Compact Disc Read Only Memory (CDROM) and the like. As such, the machine-readable storage medium 604 may include an application and/or firmware which can be utilized independently and/or in conjunction with the processing resource 602 to fetch, decode, and/or execute instructions of the machine-readable storage medium 604. The application and/or firmware may be stored on the machine-readable storage medium 604 and/or stored on another location of the computing device 600.

[0054] Although certain embodiments have been illustrated and described herein, it will be greatly appreciated by those of ordinary skill in the art that a wide variety of alternate and/or equivalent embodiments or implementations calculated to achieve the same purposes may be

substituted for the embodiments shown and described without departing from the scope of this disclosure. Those with skill in the art will readily appreciate that embodiments may be implemented in a variety of ways. This application is intended to cover adaptions or variations of the embodiments discussed herein. Therefore, it is manifestly intended that embodiments be limited only by the claims and equivalents thereof.

CLAIMS

We claim:

1. A method, executable by a networking device comprising a physical process implementing machine readable instructions, the method comprising:

detecting, using the physical processor, a specific application programming interface (API) call to interact with an application running on a production server;

assisting, using the physical processor, a scanning session of the application based on the detection of the specific API call;

identifying, using the physical processor, a modification to the application based on the scanning session.

2. The method of claim 1 comprising:

receiving, using the physical processor, an authorization of the scanning session prior to detecting the API call.

3. The method of claim 1 wherein the specific API call includes a file modification API, and wherein assisting the scanning session of the application based on the file modification API comprises:

skipping, using the physical processor, execution of the file modification API; and

reporting, using the physical processor, the skipped file modification API to the scanner.

4. The method of claim 1 wherein the specific API call includes a file modification API, and wherein assisting the scanning session of the application based on the file modification API comprises:

backing, using the physical processor, up a file affected by the file modification API;

executing, using the physical processor, the file modification API; and

reverting, using the physical processor, the file affected by the file modification API to a state prior to execution of the file modification API.

5. The method of claim 1 wherein the specific API call includes a database connection API, and wherein assisting the scanning session of the application based on the database

connection API comprises:

directing, using the physical processor, the database connection API to a testing database, the testing database is a redundant database for execution of the database connection API

6. The method of claim 1 wherein the specific API call includes a database connection API, and wherein assisting the scanning session of the application based on the database connection API comprises:

executing, using the physical processor, the database connection API;

modifying, using the physical processor, a database based on the execution of the database connection API; and

reverting, using the physical processor, the database to a state prior to the modification of the database.

7. The method of claim 1 wherein the specific API call includes an email related API, and wherein assisting the scanning session of the application based on the email related API comprises:

directing the email related API to a different email address.

8. The method of claim 1 wherein detecting the specific API call to interact with the application running on the production server comprises:

monitoring various API calls for the detection of the specific API call.

9. A system to detect a specific application programming interface (API) call for providing assistance to a scanning session, the system comprising:

a scanner to:

perform a scanning session of an application;

a production server running the application, the production server comprising:

a security agent to:

detect a specific application programming interface (API) call from the scanner;

based on the detection of the specific API call, provide assistance to the scanning session of the application; and

detect a modification to the application based on the scanning session.

10. The system of claim 9 further comprising:

a redundant resource, communicatively coupled to the production server, to execute the detected specific API call.

11. The system of claim 9 wherein the security agent is to:

authorize the scanning session; and

monitor the various API calls for the detection of the specific API call.

12. A non-transitory machine-readable storage medium comprising instructions that when executed by a physical processing resource cause a computing device to:

monitor various application programming interface (API) calls to interact with application for detection of a specific API call;

based on the detection of the specific API call, assist with a scanning session related to the detected API call; and

detect a modification to the application.

13. The non-transitory machine-readable storage medium of claim 12 further comprising instructions that when executed by the processing resource cause the computing device to:

execute the detected API call; and

revert a component to a state prior to execution of the detected API call.

14. The non-transitory machine-readable storage medium of claim 12 wherein based on the detection of the specific API call, assist with the scanning session of the application running on the server comprises instructions that when executed by the processing resource cause the computing device to:

direct execution of the API call to a redundant resource.

15. The non-transitory machine-readable storage medium of claim 12 wherein based on the detection of the specific API call, assist with the scanning session of the application running on the production server comprises instructions that when executed by the processing resource cause the computing device to:

skip execution of the API call; and

report the skipped API call to a scanner.

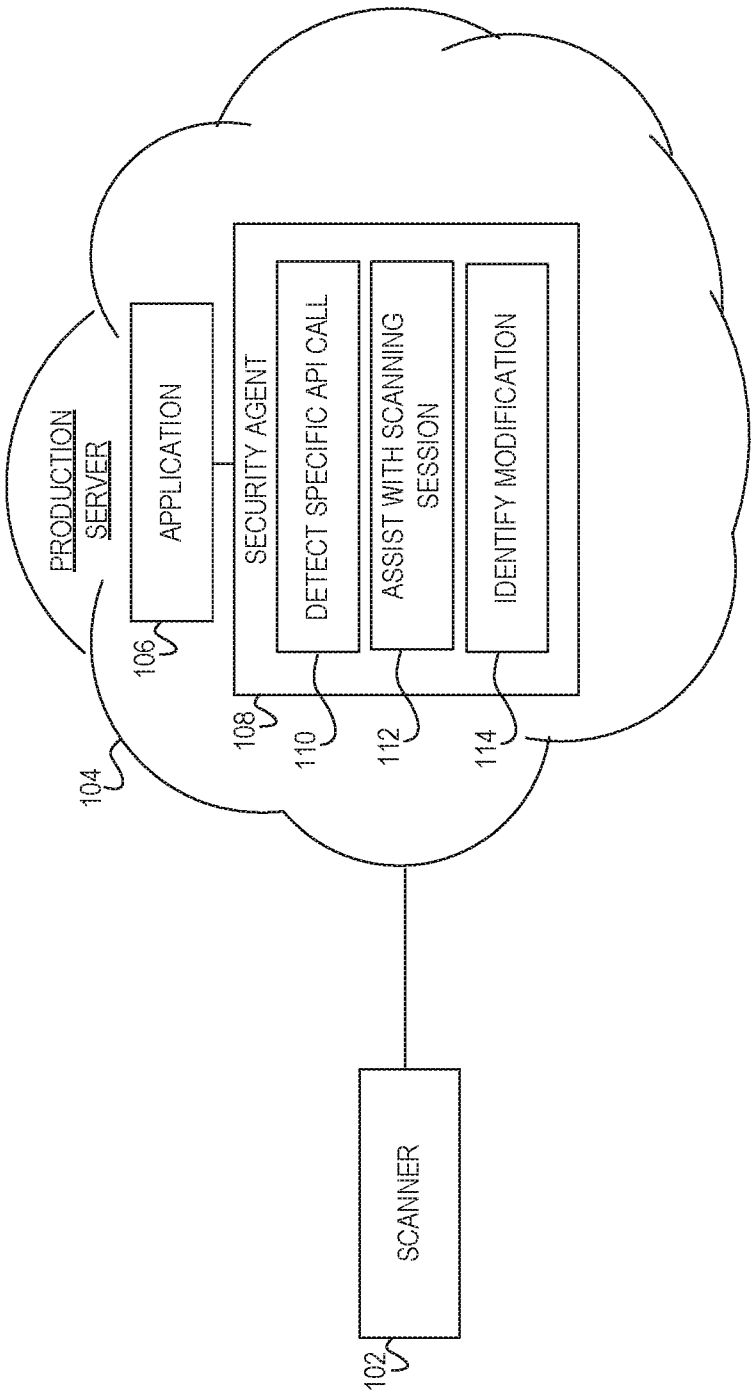

PRODUCTION SERVER
APPLICATION
106
SECURITY AGENT
DETECT SPECIFIC API CALL
ASSIST WITH SCANNING SESSION
IDENTIFY MODIFICATION
108
110
112
114
104
SCANNER
102

FIG. 1

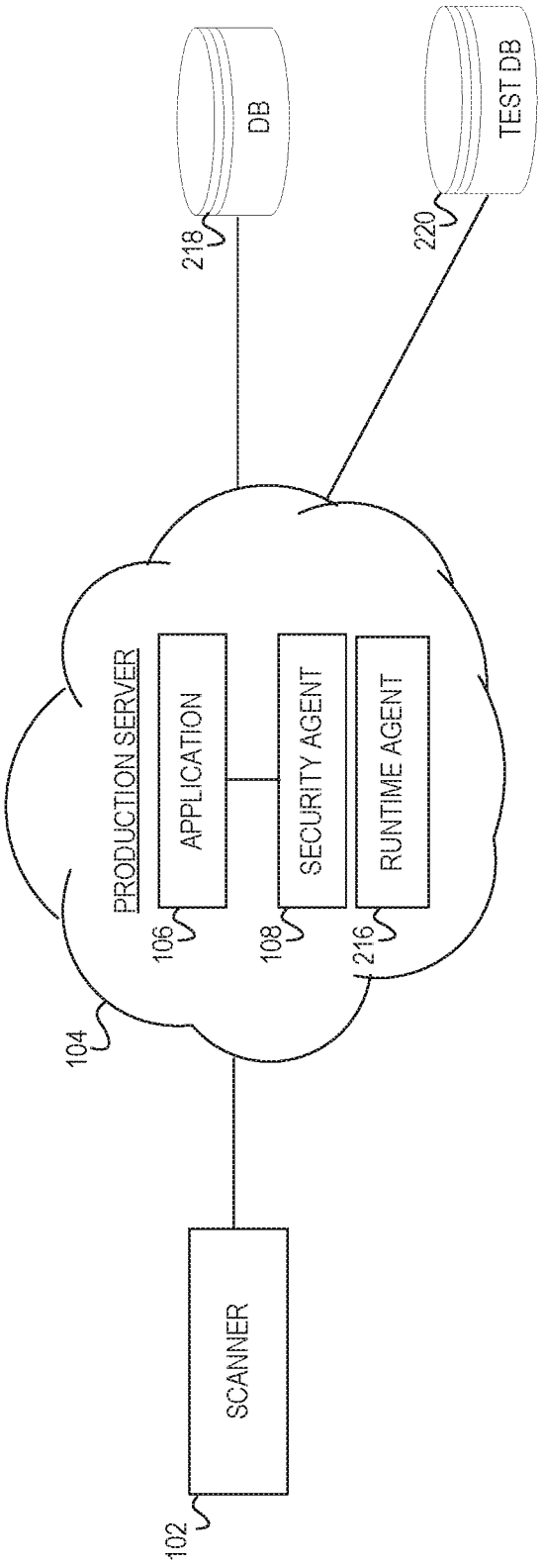

DB
TEST DB
218
220
PRODUCTION SERVER
APPLICATION
SECURITY AGENT
RUNTIME AGENT
106
108
216
104
SCANNER
102

FIG. 2

FIG. 3

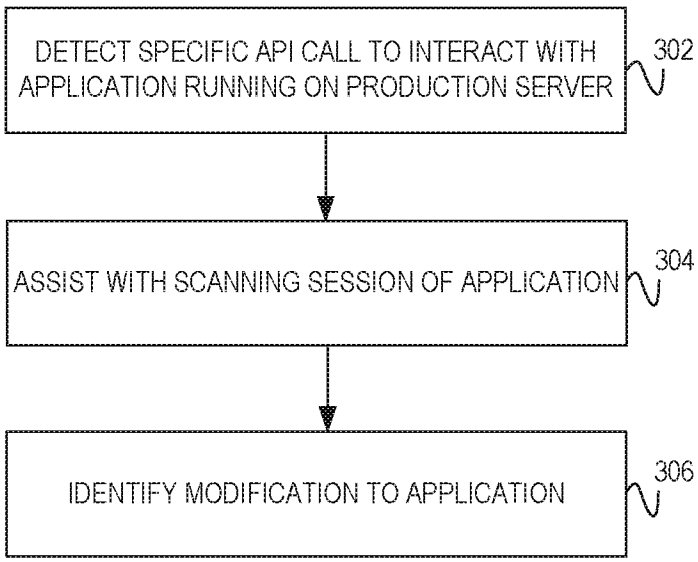
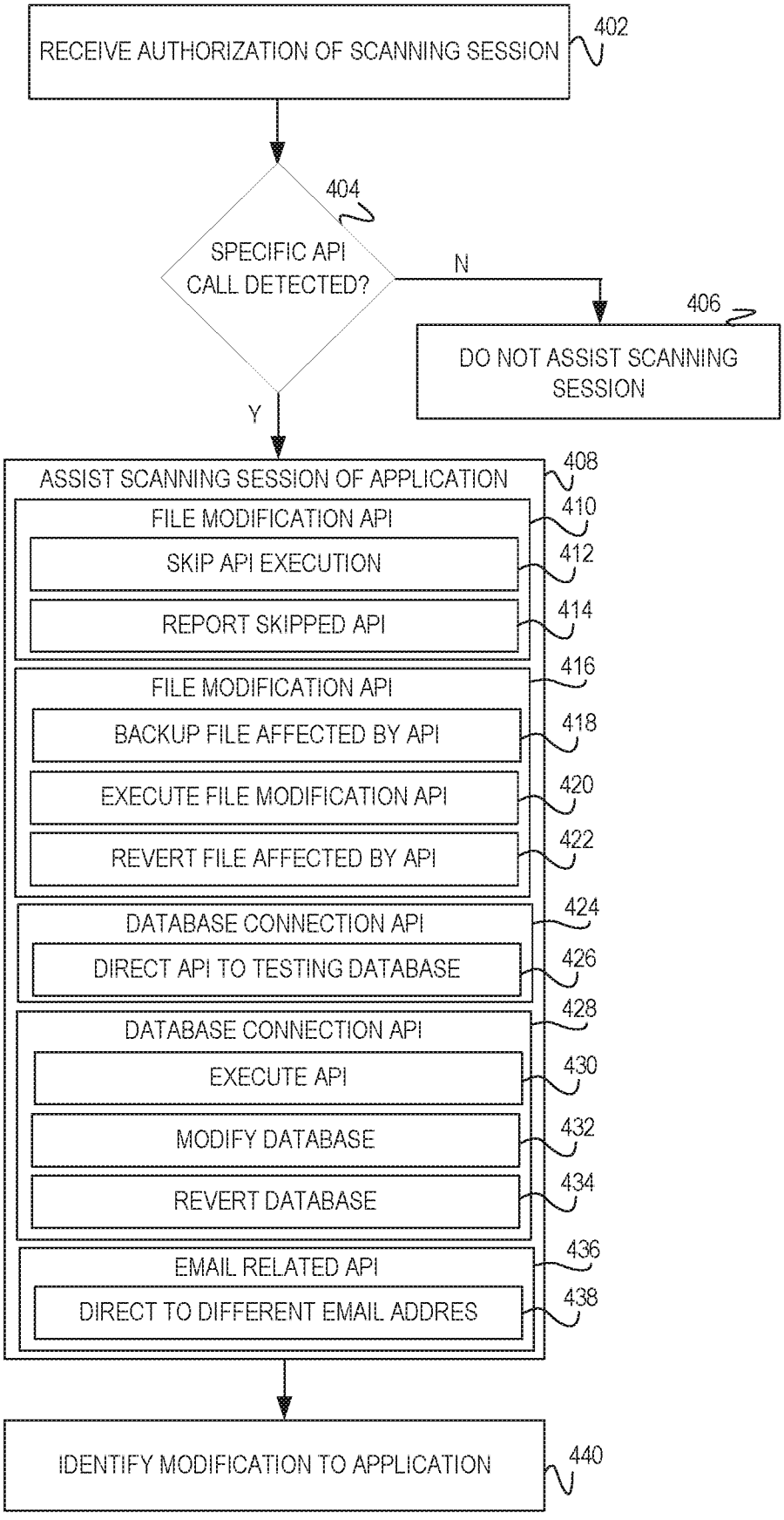

DETECT SPECIFIC API CALL TO INTERACT WITH APPLICATION RUNNING ON PRODUCTION SERVER
302
ASSIST WITH SCANNING SESSION OF APPLICATION
304
IDENTIFY MODIFICATION TO APPLICATION
306

FIG. 4


RECEIVE AUTHORIZATION OF SCANNING SESSION
402
404
SPECIFIC API CALL DETECTED?
N
406
DO NOT ASSIST SCANNING SESSION
Y
ASSIST SCANNING SESSION OF APPLICATION
408
FILE MODIFICATION API
410
SKIP API EXECUTION
412
REPORT SKIPPED API
414
FILE MODIFICATION API
416
BACKUP FILE AFFECTED BY API
418
EXECUTE FILE MODIFICATION API
420
REVERT FILE AFFECTED BY API
422
DATABASE CONNECTION API
424
DIRECT API TO TESTING DATABASE
426
DATABASE CONNECTION API
428
EXECUTE API
430
MODIFY DATABASE
432
REVERT DATABASE
434
EMAIL RELATED API
436
DIRECT TO DIFFERENT EMAIL ADDRES
438
IDENTIFY MODIFICATION TO APPLICATION
440

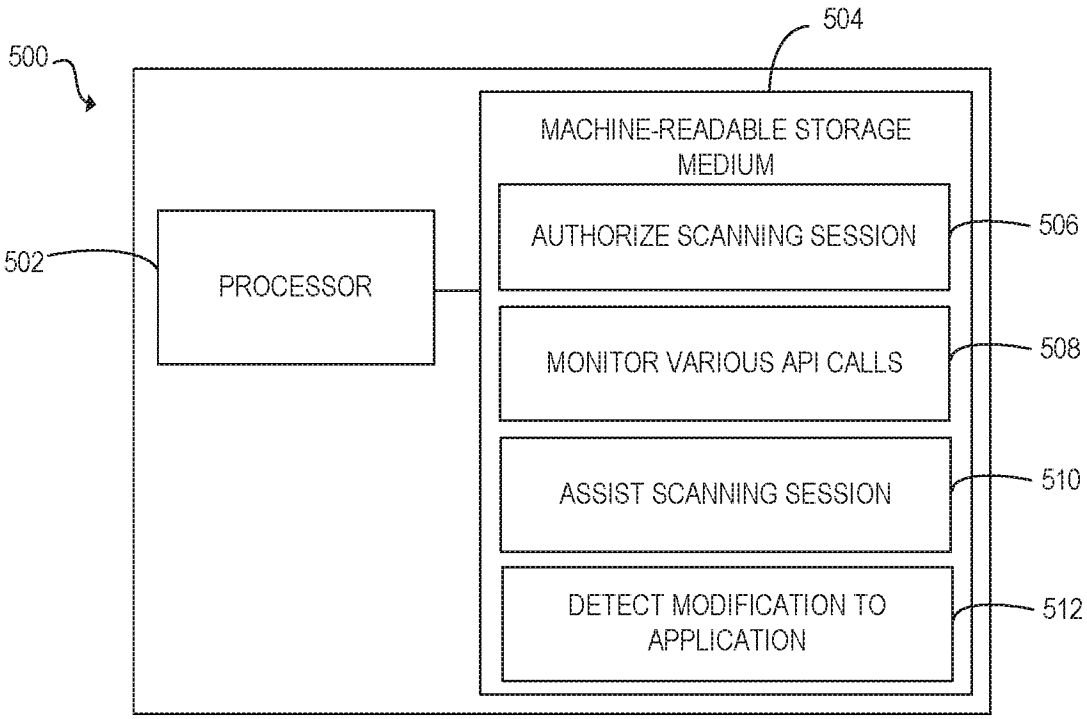

500
504
MACHINE-READABLE STORAGE MEDIUM
AUTHORIZE SCANNING SESSION
506
502
PROCESSOR
MONITOR VARIOUS API CALLS
508
ASSIST SCANNING SESSION
510
DETECT MODIFICATION TO APPLICATION
512

FIG. 5

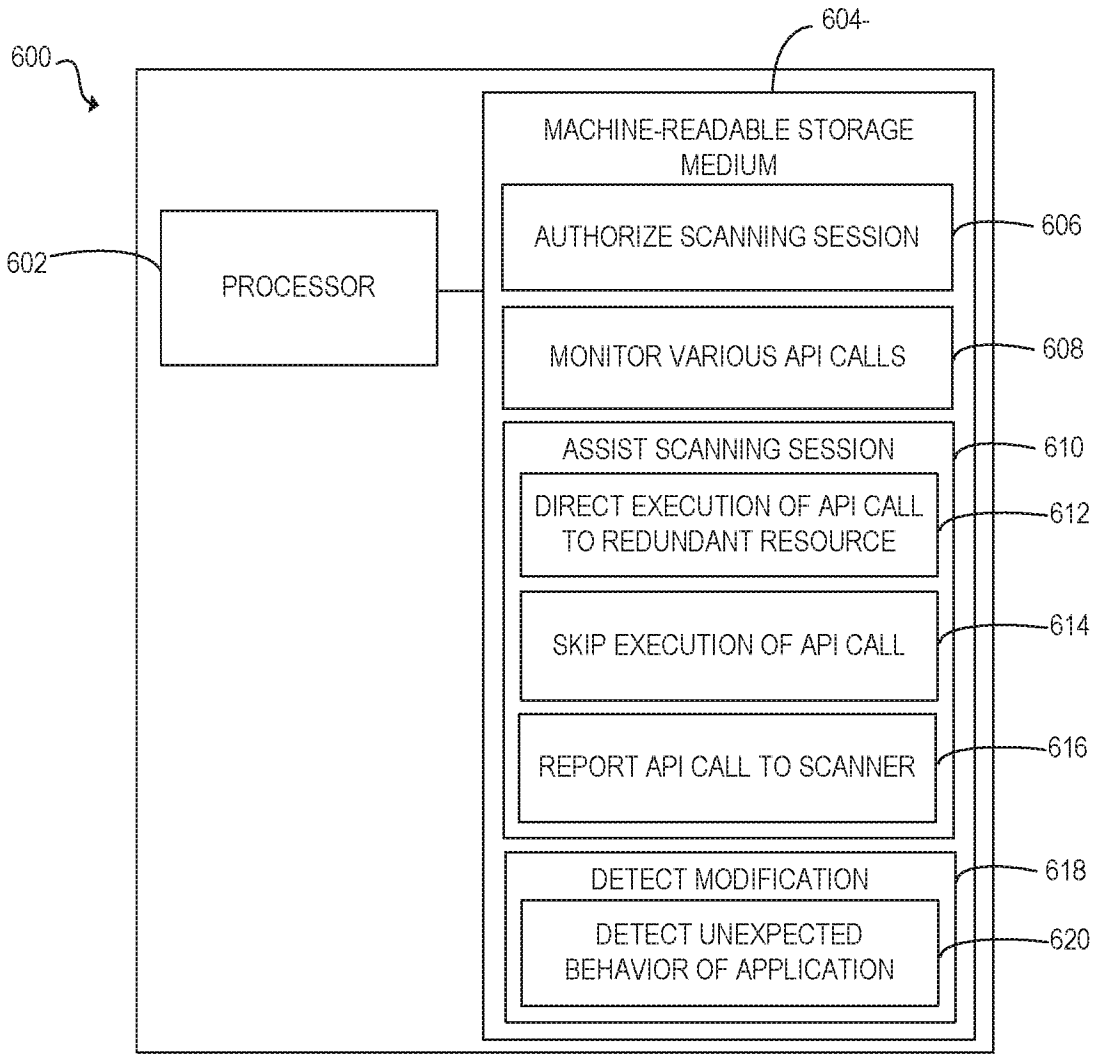

600
604
MACHINE-READABLE STORAGE MEDIUM
602
PROCESSOR
AUTHORIZE SCANNING SESSION
606
MONITOR VARIOUS API CALLS
608
ASSIST SCANNING SESSION
610
DIRECT EXECUTION OF API CALL TO REDUNDANT RESOURCE
612
SKIP EXECUTION OF API CALL
614
REPORT API CALL TO SCANNER
616
DETECT MODIFICATION
618
DETECT UNEXPECTED BEHAVIOR OF APPLICATION
620

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/023120

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/54(2006.01)i, G06F 21/50(2013.01)i, G06F 17/30(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 9/54; G06F 21/56; G06F 3/00; G06F 9/46; H04L 29/06; G06F 21/00; G06F 21/50; G06F 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: scanning session, application programming interface (API) call, email related API, file modification API, database connection API, assist

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2015-0178497 A1 (BITDEFENDER IPR MANAGEMENT LTD.) 25 June 2015 See paragraphs [0027], [0057]-[0067]; and figures 4-5B.	1-3,8-12,14-15
Y		4-7,13
Y	US 2011-0119689 A1 (DEAN CHARLES MCCRAE et al.) 19 May 2011 See paragraph [0033]; claim 4; and figure 2.	4-6,13
Y	US 2015-0026455 A1 (SYMANTEC CORPORATION) 22 January 2015 See paragraph [0029]; and figure 2.	7
A	US 2011-0307955 A1 (MARK KAPLAN et al.) 15 December 2011 See paragraphs [0027]-[0061]; and figures 1-6.	1-15
A	US 2013-0086682 A1 (LOOKOUT, INC., A CALIFORNIA CORPRATION) 04 April 2013 See paragraphs [0075]-[0105]; and figure 11.	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
06 December 2016 (06.12.2016)	**06 December 2016 (06.12.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2016/023120

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2015-0178497 A1	25/06/2015	US 9117081 B2	25/08/2015
		WO 2015-174874 A1	19/11/2015
US 2011-0119689 A1	19/05/2011	AU 2008-282721 A1	05/02/2009
		AU 2008-282721 B2	17/05/2012
		BR PI0813761 A2	30/12/2014
		CN 101772764 A	07/07/2010
		CN 101772764 B	04/07/2012
		EP 2183683 A1	12/05/2010
		EP 2183683 A4	29/08/2012
		IL 202207 D0	16/06/2010
		JP 04832595 B2	07/12/2011
		JP 2010-535388 A	18/11/2010
		RU 2010103042 A	10/08/2011
		US 2009-0037197 A1	05/02/2009
		US 7908610 B2	15/03/2011
		US 8621493 B2	31/12/2013
		WO 2009-017901 A1	05/02/2009
US 2015-0026455 A1	22/01/2015	EP 3022677 A1	25/05/2016
		US 9143497 B2	22/09/2015
		WO 2015-009508 A1	22/01/2015
US 2011-0307955 A1	15/12/2011	US 2011-0307951 A1	15/12/2011
		US 2011-0307954 A1	15/12/2011
		US 2011-0307956 A1	15/12/2011
		US 8881278 B2	04/11/2014
		US 8914879 B2	16/12/2014
		US 9081961 B2	14/07/2015
		WO 2011-156592 A2	15/12/2011
		WO 2011-156652 A1	15/12/2011
		WO 2011-156679 A1	15/12/2011
		WO 2011-156754 A1	15/12/2011
US 2013-0086682 A1	04/04/2013	US 2011-0047620 A1	24/02/2011
		US 2012-0233695 A1	13/09/2012
		US 2012-0290640 A1	15/11/2012
		US 2013-0117846 A1	09/05/2013
		US 2014-0310770 A1	16/10/2014
		US 8347386 B2	01/01/2013
		US 8544095 B2	24/09/2013
		US 8745739 B2	03/06/2014
		US 8752176 B2	10/06/2014
		US 8875289 B2	28/10/2014
		US 9294500 B2	22/03/2016